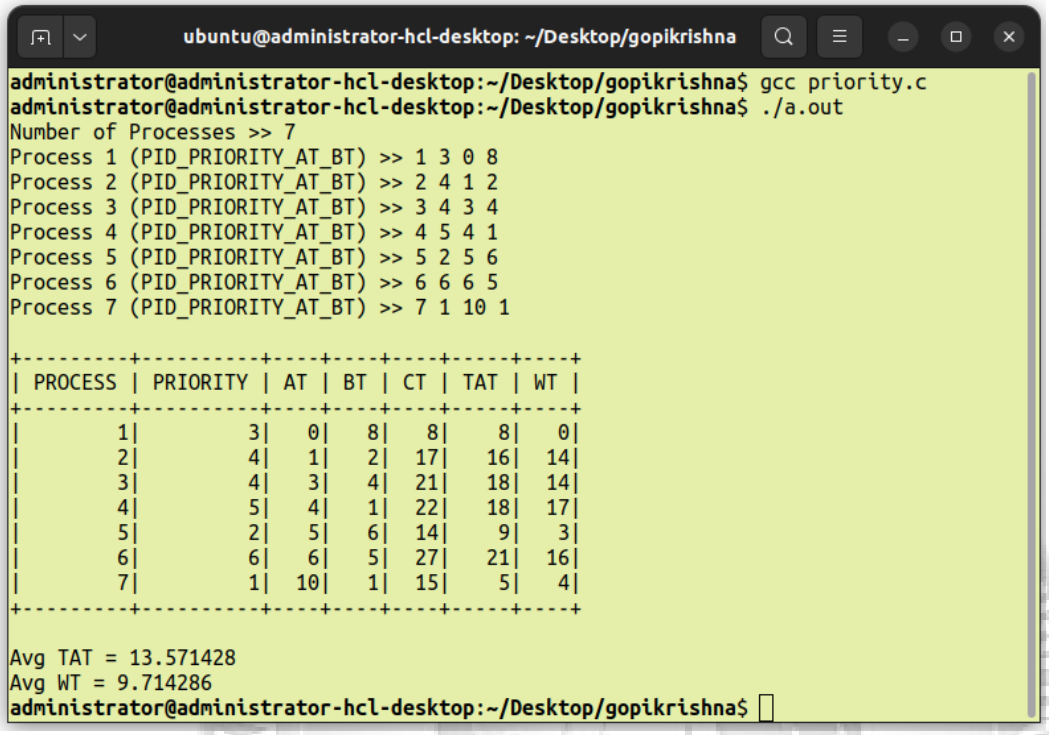
PROGRAM CODE

```
#include<stdio.h>
#include<stdlib.h>

#define STATE_UNKNOWN 0
#define STATE_READY 1
#define STATE_RETURNED 2

struct entry
{
   int AT, BT, CT, TAT, WT, ST, priority, state;
   char Name[20];
} pChart[10];
int n, readyQue[10], ready_top = 0, arrSort[10];

void swap (int* list, int i1, int i2)
{
   int temp = list[i1];
   list[i1] = list[i2];
   list[i2] = temp;
}

void enque(int id)
{
   if (pChart[id].state != STATE_UNKNOWN)
         return;
   pChart[id].state = STATE_READY;
   readyQue[ready_top] = id;
   for(int j = ready_top++; j > 0 && pChart[readyQue[j-1]].priority <=
pChart[readyQue[j]].priority; j--)
      swap(readyQue, j-1, j);
}

int nextProcessId()
{
   if (ready_top == 0) return -1;
   return readyQue[--ready_top];
}

int main ()
{
   printf("Number of Processes >> ");
   scanf("%d", &n);

   for (int i = 0; i < n; i++)
   {
      printf("Process %d (PID_PRIORITY_AT_BT) >> ", i+1);
      scanf("%s%d%d%d", pChart[i].Name, &pChart[i].priority, &pChart[i].AT, &pChart[i].BT);
      arrSort[i] = i;
      pChart[i].state = STATE_UNKNOWN;
      pChart[i].Name[7] = '\0';
```

```
}

for (int i = 1; i < n; i++)
        for (int j = i; j > 0 && pChart[arrSort[j-1]].AT > pChart[arrSort[j]].AT; j--)
                swap(arrSort, j-1, j);

int pStarted = 0, gEntry[20], gTop = 0, t_TAT = 0, t_WT = 0;
for (int cTime = 0; pStarted < n; )
{
    for (int i = 0; i < n; i++)
    {
        if (pChart[arrSort[i]].state != STATE_UNKNOWN) continue;
        if (pChart[arrSort[i]].AT > cTime) break;
        enque(arrSort[i]);
    }

    int pid = nextProcessId();
    struct entry *cp = &pChart[pid];
    if (pid > -1)
    {
        cp->ST = cTime;
        cTime += cp->BT;
        cp->CT = cTime;
        cp->TAT = cp->CT - cp->AT;
        cp->WT = cp->TAT - cp->BT;
        t_TAT += cp->TAT;
        t_WT += cp->WT;
        gEntry[gTop++] = pid;
        cp->state = STATE_RETURNED;
        pStarted++;
    }
    else
    {
        if (gEntry[gTop-1] != -1)
                gEntry[gTop++] = -1;
        cTime++;
    }
}

printf("\n+---------+----------+----+----+----+-----+----+\n");
printf("| PROCESS | PRIORITY | AT | BT | CT | TAT | WT |\n");
printf("+---------+----------+----+----+----+-----+----+\n");
for (int i = 0; i < n; i++)
{
    printf("|%9s|%10d|%4d|%4d|", pChart[i].Name, pChart[i].priority, pChart[i].AT, pChart[i].BT);
    printf("%4d|%5d|%4d|\n", pChart[i].CT, pChart[i].TAT, pChart[i].WT);
}
printf("+---------+----------+----+----+----+-----+----+\n");

printf("\nAvg TAT = %f\nAvg WT = %f\n", (float)t_TAT/n, (float)t_WT/n);
```

}

OUTPUT


ubuntu@administrator-hcl-desktop: ~/Desktop/gopikrishna
administrator@administrator-hcl-desktop:~/Desktop/gopikrishna$ gcc priority.c
administrator@administrator-hcl-desktop:~/Desktop/gopikrishna$ ./a.out
Number of Processes >> 7
Process 1 (PID_PRIORITY_AT_BT) >> 1 3 0 8
Process 2 (PID_PRIORITY_AT_BT) >> 2 4 1 2
Process 3 (PID_PRIORITY_AT_BT) >> 3 4 3 4
Process 4 (PID_PRIORITY_AT_BT) >> 4 5 4 1
Process 5 (PID_PRIORITY_AT_BT) >> 5 2 5 6
Process 6 (PID_PRIORITY_AT_BT) >> 6 6 6 5
Process 7 (PID_PRIORITY_AT_BT) >> 7 1 10 1
PROCESS	PRIORITY	AT	BT	CT	TAT	WT
1	3	0	8	8	8	0
2	4	1	2	17	16	14
3	4	3	4	21	18	14
4	5	4	1	22	18	17
5	2	5	6	14	9	3
6	6	6	5	27	21	16
7	1	10	1	15	5	4
Avg TAT = 13.571428
Avg WT = 9.714286
administrator@administrator-hcl-desktop:~/Desktop/gopikrishna$

ALGORITHM

Priority Scheduling Algorithm

- Step 0 : Start
- Step 1 : Read The number of process to 'n' (:int)
- Step 2 : Read The arrival time , burst time and priority to AT[] (:int), BT[] (:int) and PT[] (:int) respectively.
- Step 3 : Using bubble sort , sort 'AT[]' , 'BT[]' and 'PT[]' in The increasing order of the arrival time.
- Step 4 : Using bubble sort , sort 'AT[]' , 'BT[]' and PT[] except The first index of The sorted array in step 3, in The increasing order on the basis of priority.
- Step 5 : Do the following for all process , ie, i<n ; i= 0, i++
  - 5.1 : curt = curt + BT[i]
  - 5.2 : CT[i] = curt
  - 5.3 : TAT[i] = CT[i] - AT[i]
  - 5.4 : WT[i] = TAT[i] - BT[i]
  - 5.5 : ttat = ttat + TAT[i]
  - 5.6 : twt = twt + WT[i]
- Step 6 : Calculate The average TAT and WT by dividing 'ttat' and 'twt' by 'n' and then print it.
- Step 7 : Stop.